

# Compass Rose / Rosette VBA Script

Contributed by AGRC Administrator  
 27, Jul. 2006  
 Last Updated 30, Apr. 2008

This script can be used together with a UIToolControl in ArcMap to create a compass rosette in an ArcMap data view that displays a hierarchy of tick marks at 1, 5, 10 & 90 degree intervals and angle text labels at 10 and 90 degrees. To use this code, use the ArcMap customize dialog box to create a new UICommand Tool Button in the normal template and rename it "rosette". Add the tool to a toolbar. Next, open the VBA editor window within ArcMap and paste this code into the ThisDocument code window under the normal.txt item in the project explorer window.

'AGRC 07/27/2006, bgranberg

**' DESCRIPTION:**

' This script pairs with a UITool Control to creates a compass rosette in an ArcMap Data View that displays a hierarchy of tick marks at 1, 5, 10 & 90 degree intervals and angle text labels at 10 and 90 degrees. To use this code, use the ArcMap customize dialog box to create a new UICommand Tool Control in the normal template and rename it 'rosette'. Add the tool to a toolbar. Next, open the VBA editor window within ArcMap and paste this code into the ThisDocument code window under the normal.txt item in the project explorer window.

**' REQUIREMENTS:**

' &mdash; ArcMap

**' NOTES:**

' &mdash; This script will create 396 graphic elements, group them before you do anything  
 ' &mdash; The center of the rosette is defined with the mousedown event when using the rosette tool  
 ' &mdash; The radius is defined as the distance between the mousedown and mouseup events.

```
Private Sub Rosette_Select()
```

```
    MsgBox "Click Center and Drag Radius to Create Compass Rosette", vbOKOnly, "Rosette Tool"
End Sub
```

```
Private Sub Rosette_MouseDown(ByVal button As Long, ByVal shift As Long, ByVal x As Long, ByVal y As Long)
```

```
    Dim pMxApp As IMxApplication
    Dim pMxDoc As IMxDocument
    Dim pCirArc As ICircularArc
    Dim pPoint As IPoint
    Dim pRubberCircle As IRubberBand
```

```
    Set pMxDoc = ThisDocument
    Set pMxApp = Application
```

```
    Set pRubberCircle = New RubberCircle
```

```
    ' Return a new circle from the tracker object using TrackNew
    Set pCirArc = pRubberCircle.TrackNew(pMxDoc.ActiveView.ScreenDisplay, Nothing)
    Call DrawRosette(pCirArc.CenterPoint, pCirArc.Radius)
```

```
End Sub
```

```
Private Sub DrawRosette(inPt As IPoint, inRadius As Double)
```

```

Dim pMxApp As IMxApplication
Dim pMxDoc As IMxDocument
Dim pLine As ILine
Dim pElement As IElement
Dim pGC As IGraphicsContainer
Dim pAV As IActiveView
Dim pSegmentCollection As ISegmentCollection

```

```

Set pMxApp = Application
Set pMxDoc = Application.Document
Set pGC = pMxDoc.FocusMap

```

```

Dim d As Integer
Dim centerX As Double
Dim centerY As Double
Dim centerPt As IPt
Dim outerPt As IPt
Dim outerConPt As IConstructPoint
Dim innerPt As IPt
Dim innerConPt As IConstructPoint
Dim pTextPt As IPt
Dim pTextConPt As IConstructPoint
Dim pTextPt2 As IPt
Dim pTextConPt2 As IConstructPoint
Dim pTextLine As ILine
Dim tickLine As ILine
Dim degree90Ratio As Double
Dim degree10Ratio As Double
Dim degree5Ratio As Double
Dim degree1Ratio As Double

```

```

Dim pLineElement As ILineElement
Dim pTextElement As ITextElement
Dim pLineSymbol As ILineSymbol
Dim pRGBColor As IRgbColor
Dim degree90TextRatio As Double
Dim degree10TextRatio As Double
Dim t2d As ITransform2D

```

```

'ratios along radius line for different degree increments
degree90Ratio = 0.9
degree10Ratio = 0.95
degree5Ratio = 0.97
degree1Ratio = 0.985
degree90TextRatio = 0.82
degree10TextRatio = 0.88

```

```

centerX = inPt.x
centerY = inPt.y

```

```

Set centerPt = New Point

```

```

centerPt.x = centerX
centerPt.y = centerY
Set outerPt = New Point
Set outerConPt = outerPt
Set innerPt = New Point
Set innerConPt = innerPt
Set pTextPt = New Point
Set pTextPt2 = New Point
Set pTextConPt = pTextPt
Set pTextConPt2 = pTextPt2
Set pLine = New Line
Set pTextLine = New Line

```

```

For d = 0 To 359
If CInt(d / 90) = d / 90 Then

    outerConPt.ConstructAngleDistance centerPt, (450 - d) * 3.14159 / 180, inRadius
    innerConPt.ConstructAngleDistance centerPt, (450 - d) * 3.14159 / 180, inRadius * degree90Ratio
    pLine.FromPoint = outerConPt
    pLineToPoint = innerConPt

    'add line to the Segmentcollection
    Set pSegmentCollection = New Polyline
    pSegmentCollection.AddSegment pLine

    'add polyline to the graphicscontainer
    Set pElement = New LineElement
    Set pLineElement = pElement

    pElement.Geometry = pSegmentCollection
    Set pLineElement = pElement 'QI
    Set pLineSymbol = pLineElement.Symbol
    Set pRGBColor = New RgbColor
    pRGBColor.Blue = 0
    pRGBColor.Red = 0
    pRGBColor.Green = 0
    pLineSymbol.Color = pRGBColor
    pLineElement.Symbol = pLineSymbol
    pGC.AddElement pElement, 0

    pTextConPt.ConstructAngleDistance centerPt, (450 - d) * 3.14159 / 180, inRadius * degree90TextRatio
    Set pTextElement = MakeATextElementFromPoint(pTextPt, CStr(d), 16, CDbl(d))

    Set t2d = pTextElement
    t2d.Rotate pTextPt, (360 - d) * 3.14159 / 180

    Debug.Print CStr(d) & " " & 16 & " " & d
    pGC.AddElement pTextElement, 0

    'Debug.Print outerPt.x & " " & outerPt.Y & " ---" & innerPt.x & " " & innerPt.Y

Elseif CInt(d / 10) = d / 10 Then

    'Debug.Print "m10 " & d

    outerConPt.ConstructAngleDistance centerPt, (450 - d) * 3.14159 / 180, inRadius
    innerConPt.ConstructAngleDistance centerPt, (450 - d) * 3.14159 / 180, inRadius * degree10Ratio
    pLine.FromPoint = outerConPt
    pLineToPoint = innerConPt

    'add line to the Segmentcollection
    Set pSegmentCollection = New Polyline
    pSegmentCollection.AddSegment pLine

    'add polyline to the graphicscontainer
    Set pElement = New LineElement
    Set pLineElement = pElement

    pElement.Geometry = pSegmentCollection
    Set pLineElement = pElement 'QI
    Set pLineSymbol = pLineElement.Symbol
    Set pRGBColor = New RgbColor
    pRGBColor.Blue = 51
    pRGBColor.Red = 51
    pRGBColor.Green = 51
    pLineSymbol.Color = pRGBColor
    pLineElement.Symbol = pLineSymbol

```

```

pGC.AddElement pElement, 0

pTextConPt.ConstructAngleDistance centerPt, (450 - d) * 3.14159 / 180, inRadius * degree90TextRatio
Set pTextElement = MakeATextElementFromPoint(pTextPt, CStr(d), 8, CDbl(d))

Set t2d = pTextElement
t2d.Rotate pTextPt, (360 - d) * 3.14159 / 180

Debug.Print CStr(d) & " " & 8 & " " & d
pGC.AddElement pTextElement, 0

ElseIf CInt(d / 5) = d / 5 Then

'Debug.Print "s5 " & d

outerConPt.ConstructAngleDistance centerPt, (450 - d) * 3.14159 / 180, inRadius
innerConPt.ConstructAngleDistance centerPt, (450 - d) * 3.14159 / 180, inRadius * degree5Ratio
pLine.FromPoint = outerConPt
pLineToPoint = innerConPt

'add line to the Segmentcollection
Set pSegmentCollection = New Polyline
pSegmentCollection.AddSegment pLine

'add polyline to the graphicscontainer
Set pElement = New LineElement
Set pLineElement = pElement

pElement.Geometry = pSegmentCollection
Set pLineElement = pElement 'QI
Set pLineSymbol = pLineElement.Symbol
Set pRGBColor = New RgbColor
pRGBColor.Blue = 102
pRGBColor.Red = 102
pRGBColor.Green = 102
pLineSymbol.Color = pRGBColor
pLineElement.Symbol = pLineSymbol
pGC.AddElement pElement, 0

Else

outerConPt.ConstructAngleDistance centerPt, (450 - d) * 3.14159 / 180, inRadius
innerConPt.ConstructAngleDistance centerPt, (450 - d) * 3.14159 / 180, inRadius * degree1Ratio
pLine.FromPoint = outerConPt
pLineToPoint = innerConPt

'add line to the Segmentcollection
Set pSegmentCollection = New Polyline
pSegmentCollection.AddSegment pLine

'add polyline to the graphicscontainer
Set pElement = New LineElement
Set pLineElement = pElement

pElement.Geometry = pSegmentCollection
Set pLineElement = pElement 'QI
Set pLineSymbol = pLineElement.Symbol
Set pRGBColor = New RgbColor
pRGBColor.Blue = 151
pRGBColor.Red = 151
pRGBColor.Green = 151
pLineSymbol.Color = pRGBColor
pLineElement.Symbol = pLineSymbol
pGC.AddElement pElement, 0

```

```
End If
```

```
Next d
```

```
'refresh activeview
Set pAV = pMxDoc.FocusMap
pAV.PartialRefresh esriViewGraphics, Nothing, Nothing
```

```
End Sub
```

```
Public Function MakeATextElementFromPoint(pPoint As IPoint, strText As String, textSize As Double, textAngle As Double) As IElement
```

```
'Pointers needed to make text element
Dim pRGBColor As IRgbColor
Dim pTextElement As ITextElement
Dim pTextSymbol As ITextSymbol
Dim fnt As IFontDisp
Dim pElement As IElement
```

```
'First setup a color. We'll use RGB red
Set pRGBColor = New RgbColor
pRGBColor.Blue = 0
pRGBColor.Red = 0
pRGBColor.Green = 0
```

```
'Next, cocreate a new TextElement
Set pTextElement = New TextElement
'Query Interface (QI) to an IElement pointer and set
'the geometry that was passed in
Set pElement = pTextElement
pElement.Geometry = pPoint
```

```
'Next, setup a font
Dim pFontDisp As IFontDisp
Set pFontDisp = New stdole.StdFont
pFontDisp.Name = "Arial"
pFontDisp.Bold = False
pFontDisp.Underline = False
```

```
'Next, setup a TextSymbol that the TextElement will draw with
Set pTextSymbol = New TextSymbol
pTextSymbol.Font = pFontDisp
pTextSymbol.Color = pRGBColor
pTextSymbol.size = textSize 'set the size of the text symbol here, rather than on the font
pTextSymbol.VerticalAlignment = esriTVACenter
pTextSymbol.HorizontalAlignment = esriTHACenter
pTextSymbol.Angle = textAngle
'Debug.Print pTextSymbol.Angle
```

```
'Dim pFormat As IFormattedTextSymbol
'Set pFormat = pTextSymbol
'pFormat.Kerning = False
'pFormat.Direction = esriTDVertical
'pFormat.Angle = textAngle
'pFormat.Text = strText
```

```
'Next, Give the TextSymbol and text string to the TextElement
pTextElement.Symbol = pTextSymbol
pTextElement.Text = strText
```

```
'Finally, hand back the new element as the output of this function
```

Set MakeATextElementFromPoint = pTextElement

End Function